International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Journal Européen des Systèmes Automatisés
Vol. 56, No. 6, December, 2023, pp. 1043-1051
Journal homepage: http://iieta.org/journals/jesa

Software Quality Assessment Technique for the Autonomous Power Plants Automated Control Systems

Mahmoud M. S. Al-suod[1*], Oleksandr Ushkarenko[2], Olha Dorohan[2], Abdullah Eial Awwad[1], Alaa Al-Quteimat[1]

[1] Department of Electrical Power Engineering and Mechatronics, Tafila Technical University, Tafila 66110, Jordan
[2] Department of Programmable Electronics, Electrical Engineering and Telecommunications, Admiral Makarov National University of Shipbuilding, Mykolaiv 54050, Ukraine

Corresponding Author Email: m.alsoud@ttu.edu.jo

https://doi.org/10.18280/jesa.560614

Received: 4 September 2023
Revised: 8 November 2023
Accepted: 4 December 2023
Available online: 28 December 2023

Keywords:
software quality, testing, user interface, peer review technique, quality attributes

ABSTRACT

The paper proposes further development of formal method for obtaining the values of software quality attributes and assessing the quality of software used for controlling and parameters monitoring of autonomous electric power systems. For this purpose, dynamic software testing was used with the involvement of a group of experts. The results of the conducted expert assessment were used as initial data. The attributes of software quality indicators, such as functionality, practicality, maintainability, reliability, were calculated using the method of summarizing and grouping the results of statistical observation that allowed to check the software compliance with quality standards. Additional weighting coefficients that describe the importance of individual software quality attributes were introduced, and additive convolution is used to calculate the values of various software quality indicators. Minimax criterion was used to find the best solution that maximizes the quality of the software and minimizes possible losses due to errors. The technique proposed in the paper makes it possible to obtain quantitative assessments of software quality based on statistical processing of testing results and expert assessments. This allows to select specific software characteristics for improvement without affecting others, to predict software failureless time and to minimize the subjective factor during testing.

1. INTRODUCTION

Software for monitoring parameters and controlling operating modes of autonomous electric power systems (AEPS) is part of automated control systems (ACS) and plays an important role in ensuring the process of energy-efficient generation and distribution of electricity. The purpose of the top-level software discussed in this paper is parameter monitoring and remote control of AEPS. Such software, in a sense, can be considered as a human-machine interface, and the AEPS is controlled through user interaction with on-screen controls. Therefore, to ensure reliable and uninterrupted operation of the entire AEPS, an important factor is to ensure reliable operation of the software and its compliance with functional requirements, achieved by identifying and correcting errors in the algorithms at the testing stage or directly during operation. At the same time, in this process, an important role plays the practicality of use and understandability of the program interface for users, the duration of training, the convenience of controlling the operating modes of individual elements of the AEPS, the accuracy of reading data about the operating mode of the AEPS and the state of its individual elements from the monitor screen using text or special graphical interface elements.

During the life cycle of the software, changes are made to it, new functionalities are added, and appearance of both individual graphic elements and the entire program interface as a whole change. In particular, in the study [1], the influence of different refactoring approaches on quality attributes of software is demonstrated. And such changes, although they are aimed at improving the software quality, can have an opposite effect, namely bring errors into the work logic of the program, worsen the usability, reduce understandability of the program interface for users, which will have a negative impact on the operation of the entire autonomous electric power system (AEPS). The software testing process helps to prevent this, or at least lessen the likelihood. But, despite the fact that there are many successful examples, the problem of assessing the quality of the SW itself has not been fully resolved to date and still relevant [2]. In particular, there is a lack of formal approach for assessing the usability of software for AEPS control and parameters monitoring, which considers the degree of importance of various software quality indicators, while minimizing the subjective factor when using expert evaluation. Analyzing all aspects of this issue, one can make a conclusion that the problem may be caused both by the existing understanding of the term "quality of software" itself, and by the techniques that are used to calculate various quality indicators [3, 4]. The authors come to such a conclusion not only based on the analysis of regulatory documents and specialized literature [5, 6], but also based on repeated participation in the acceptance-passing tests of ACS AEPS.

The software for AEPS control considered in this article is a specialized SCADA system designed by the authors, taking into account the requirements described in Lyaskovskiy et al. [7]. The structure of the software in the form of a pattern network for analysis of information flows for compliance of the software with the requirements of operation and control of the equipment in real time is given in Mahmoud et al. [8], however, the issue of testing this software was not considered. In Al-Suod et al. [9], the method of functional software testing has been considered, which is only a separate aspect of software quality analysis. In addition, the values of quantitative indicators of other quality attributes, which would allow a more formal evaluation of the quality of the software under consideration, were not given. This is one of the research gaps. Moreover, the presence of numerical values of individual quality attributes allows using software quality prediction methods, as shown in Hovorushchenko et al. [10]. Traditional testing approaches are manual testing [11] and automated unit testing [12]. Manual testing involves testers performing tests manually by interacting with the software, while automated testing involves using software tools to perform tests automatically. Usage of these approaches, of course, allows you to detect errors in the operation of the software, but it has its drawbacks. In general, testers are not end users of the software and may not fully understand the features of AEPS operation and evaluate the convenience and understandability of the AEPS management process. In addition, manual testing requires availability of a working ACS AEPS, and the work of testers should take place on it. However, in this case, the testing process itself will be expensive from an economic point of view, because, firstly, it requires availability of working equipment, and secondly, creation of various emergency situations, and in case of failures in operation of the tested software, this can lead to damage to the equipment, and, as a result, significant economic costs. The automated unit testing, as shown in [12], does reduce the density of software defects, but only in some cases provides 100% coverage of the software code with tests, provided that the developed program is relatively simple, but the SCADA systems are not simple. Also, the unit testing does not provide information about such attributes of software quality as ease of use, practicality, ease of learning, and some others, as stated in the study [13]. Therefore, the peer review technique is also used to assess the software quality. Although this method is quite popular and, as shown in Eisty and Carver [14], allows designers to conduct comprehensive testing of the software code. However, in the study [14], we are talking about expert verification of the code for compliance with the best modern practices of software development. However, in scientific publications, the peculiarities of using the peer review technique for testing the software of the ACS AEPS, namely, selection of quality attributes for testing and further processing of the received data, are insufficiently described, which is also a research gap. Thus, the research fills the gap of the lack of methodology for a formal approach to assessing software usability, which will allow developers to select specific software characteristics for improvement without affecting others, predict the trouble-free operation of the software and minimize the subjective factor during testing.

2. MODELING AND ANALYSIS

The aim of the research is to develop the methodology for determining software quality by using the peer review technique to obtain data and calculate using statistical analysis methods of quantitative values of the most important quality attributes of specially designed software for monitoring parameters and managing the autonomous electric power system and studying its compliance with quality standards.

2.1 A general description of the software for monitoring parameters and controlling of AEPS

The software as a component of ACS AEPS according to the incremental model of development of a complicated system at the initial stage can be considered as an independent component. Then its creation is associated with the following processes: definition of requirements, SW designing and construction, quality assurance, documentation, and configuration management.

As shown in Figure 1, the main window of software tools for managing and monitoring AEPS parameters is presented.

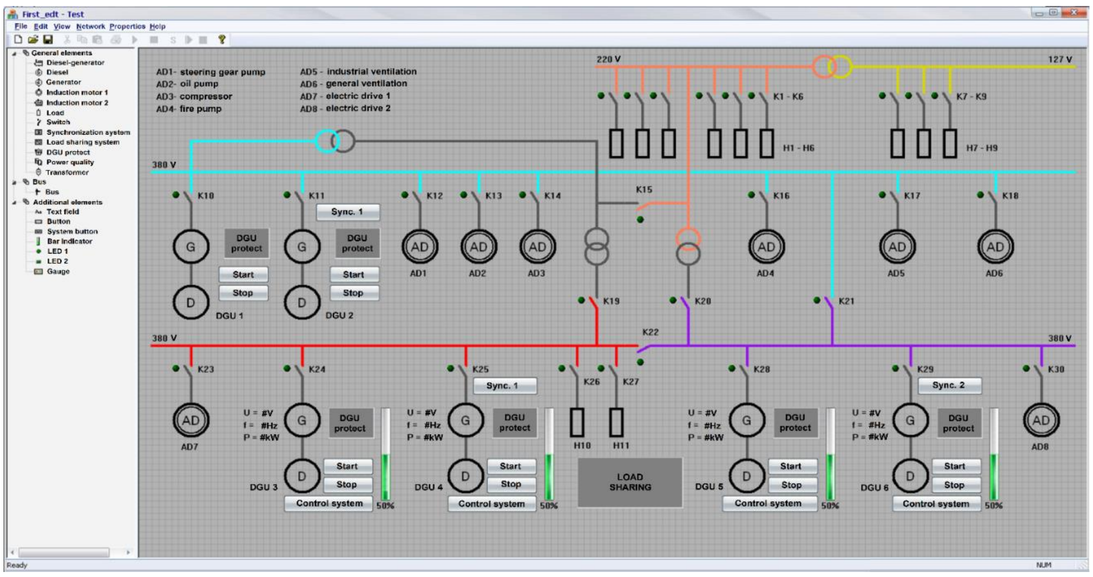

Figure 1. The main window of the application for the autonomous electric power system control

Software tools allows to display both the structure of the system and the state of separate elements (subsystems) and devices. Therefore, according to its structure, the software tools for monitoring and control AEPS are multi dialog application, in which the feature of quick access to additional information has been implemented by opening additional dialog windows. Using mnemonic schemes is a common approach for dispatcher control panels that in modern systems are executed on an industrial computer and provides for the operator the information about the current state of the AEPS.

The purpose of the top-level software is remote monitoring and control of AEPS. To implement these functions, the software has the means to create a mnemonic scheme of AEPS, parameters monitoring and control of AEPS in real time, analysis the operating modes of power units. Software requirements can be divided into three groups according to the actors involved in individual stages of work - System Setup Operator (creates a mnemonic scheme, configures and adjusts the properties of its components), Power System Control Operator (performs real-time monitoring and control of the power system) and Analyst (performs analysis of operating modes of software and AEPS in order to optimize them).

The main actor in the system is the Power System Control Operator. The actions perform by them determine the requirements for the component library and user interface and are the basis for collecting data used to analyze the operation of the whole system.

The preparatory stage before starting the process of monitoring the parameters and controlling the AEPS in real time is the creation of a mnemonic scheme of the electric power system and setting the properties of its components.

The structure of mnemonics is based on a few principles that should be followed in the process of developing the software: the principle of brevity (the scheme should be simple and without redundant elements, information must be shown clearly and in a user-friendly form), the principle of generalization (the most essential features of the managed facilities are highlighted), the principle of emphasis (control elements can be distinguished by their size, color, shape etc.), the principle of dimensional correlation of control and management elements, location of control and measuring and indicator devices, the principle of using intimate associations.

To perform the necessary actions, the System Setup Operator has the ability to interact with the library of components to create them by moving components from the library to the working field using Drag and Drop technique. Each graphical component supports move (using the mouse cursor and keyboard arrows) and delete operations. A common property of the software environment at the stage of preparing the system for operation in the AEPS control mode is the ability to set the properties of components and the data exchange channel for information interaction with automation hardware, create connections between components to monitor parameters in a form convenient for the Operator, change the color scheme of the environment and capabilities saving/restoring a created and configured mnemonic scheme.

When the software operates in the AEPS monitoring and control mode, in addition to performing basic actions, information is collected about the operation of the program (load of the data exchange channel, content of information packages) and power units for a certain time. The stored information is then processed by the Analyst.

1. Verification of the previously calculated system response time and decisions on the possibility of using protection for diesel generator units (DGU) and controlling discrete signals at the software level; for these purposes, information about the communication channel load is used.

2. Identifying and eliminating malfunctions when exchanging data with automation tools (data on the contents of information packages is used).

3. Calculation of individual indicators of power quality (average, minimum and maximum values and frequencies of power dips and surges) based on information about the operating modes of the DGUs.

4. Analysis of operating modes of generating units to formulate a AEPS control strategy in which the amount of fuel consumed by the diesel generator will be minimal, for example, by reconfiguring the AEPS structure.

The Analyst's tasks also include conducting functional software testing. Since using the software is a three-stage iterative process (changing or creating mnemonic scheme of a power plant and setting the properties of its components, using the program in the main mode, analyzing the operation of the system), both the first and third stages are visually identical from the user's point of view, and the process of using the software can be divided into two modes: circuit designer mode and AEPS monitoring and control mode.

Software quality requirements are regulated by the Square software quality model, based on ISO/IEC (25000-25099) standards [15, 16]. According to this model, software quality assessment is carried out by considering its compliance with requirements divided into six categories: functionality, practicality, efficiency, reliability, maintainability and mobility. In the studies [17-19] the methods for evaluating various software quality characteristics are given, based on which analysis for evaluation of quality of software for monitoring and managing AEPS it is suggested to use testing to evaluate functionality, expert survey to evaluate practicality, and analytical methods to evaluate other characteristics.

2.2 The research of compliance of the software with the quality standards

When designing the top-level software of the ACS AEPS, one of the key issues is its quality. The Square software quality model, based on ISO/IEC (25000-25099) standards, is recognized as generally accepted. In accordance to this model, quality assessment is a three-level and involve determining the necessary characteristics for each type of software, for each of the characteristics – attributes and for each of the attributes – metrics. The standards also regulate the main characteristics for all types of software: functionality, practicality, efficiency, reliability, maintainability and mobility. Below are the methods of calculating the main characteristics for the designed software.

As attributes of the functionality of the software, it is possible to distinguish the suitability and accuracy of the performance of functions, the ability to interact with third-party software and automation hardware [9]. In Dougherty et al. [20], the methods of evaluating software functionality and the justified effectiveness of using testing for this purpose are presented.

The basis for software functionality testing is the requirements it must meet. Since the software is an inalienable part of the software-hardware complex, its testing requires the presence of models simulating the behavior of the hardware and the automation facility as a whole.

Functional tests are usually divided into several stages:

modular, integration and system [21]. The testing procedure consists in performing the following sequence of actions: determination of the test fullness criterion; compilation of a full set of test situations; drawing up a report with the test results. The results of tests should be represented as the values of the metrics of the attributes of functionality – suitability, accuracy, and interoperability. Since software testing is a multi-stage process, Table 1 was created, which was filled in after passing each stage of testing.

In Table 1, the numbers in the first column correspond to the following visual metaphors (components and means of analysis).

The aim of unit tests is to determine the metric of the separate software components functionality attributes. For most software components, behavior in designer mode significantly varies from behavior in power plant control mode and is implemented (designed and coded) in the form of separate finite-state machines. The automatic approach allows you to isomorphically switch to an automaton that models behavior of the component, and as a criterion for the completeness of testing, use the coverage of its transitions [22]. Usage of two machines permits to detach the computation of suitability metrics (fullness of the test coverage the machine that simulates the behavior in the scheme designer mode) and interoperability (for the behavior in the power system control mode). A similar approach is acceptable only for components involved into data exchange with automation hardware and do not require mandatory information exchange with other components (DGA, DGA Protection, Power Quality Synchronization and Monitoring System, Control Button, Automatic Switch, LED). For components that are used only for the visual completeness of the mnemonic and do not require communication with other components (Asynchronous Motor, Transformer, All Display Components), only suitability is determined. The value of accuracy as the degree of compliance of device operation with the set limit value at the stage of modular testing can be calculated only for the DGA Protection component. Thus, as a criterion for completeness of modular testing of all components for which it is carried out, completeness of coverage of automatic machine transitions is determined; a complete set of test situations is created based on the transition conditions of machines. Since this testing stage refers to the stage of debugging, the transition to adding functionality and conducting the further stage of testing occurs only under the condition of 100% coverage of test situations.

The final stage of testing is a system test. The operation of a power plant is a certain sequence of operation of the DGUs and actuators in accordance with the technological process. The purpose of system testing is to check the functionality of the software when controlling a power plant, therefore, the coverage of software requirements can be highlighted as a criterion for the completeness of testing. Before compiling a set of test cases, it is necessary to determine the classes of nonequivalent situations. It is proposed to use the following set of non-equivalent situations: using a component of each type in the mnemonic scheme designer mode; start and stop of DGU; connecting and disconnecting load from the main switchboard buses; display of DGU parameters; activation of each of the DGU protection functions; synchronization of the DGU with the main switchboard buses; load sharing between parallel operating DGUs. To fully test the software, the set of situations must include all of the listed non-equivalent situations.

Table 1. The results of functional software testing

Visual Metaphor (Components and Means of Analysis)	**Modular Testing**			**Integration Testing**			**System Testing**								
							Suitability			**Ability**			**Precision**		
	Suitability	**Ability**	**Precision**	**Suitability**	**Ability**	**Precision**	**Weight Factor**	**Full Coverage**	**Test Results**	**Weight Factor**	**Full Coverage**	**Test Results**	**Weight Factor**	**Full Coverage**	**Test Results**
Diesel- generator unit	1	1	–	1	1	–	1	1	1	0.5	1	1	–	–	–
Protection of the DGU	1	1	1	1	1	–	1	1	1	1	1	1	1	1	0.9
Synchronization system	1	1	–	1	1	–	1	1	1	1	1	1	–	–	–
Load sharing system	–	–	–	1	1	1	1	0.7	1	1	1	1	0.8	1	0.7
Power quality monitoring system	1	1	–	–	–	–	1	1	1	–	–	–	–	–	–
Automatic switch	1	1	–	1	1	–	1	1	1	1	1	1	–	–	–
LED	1	1	–	–	–	–	1	1	1	0.5	1	1	–	–	–
Latch button	1	1	–	–	–	–	1	0	0	1	0	0	–	–	–
Non-latch button	1	1	–	–	–	–	1	1	1	1	1	1	–	–	–
Indication components	–	–	–	1	–	–	1	1	1	–	–	–	–	–	–
Induction motor	1	–	–	1	–	–	1	1	1	–	–	–	–	–	–
Transformer	1	–	–	1	–	–	1	1	1	–	–	–	–	–	–
Bus	–	–	–	1	–	–	1	1	1	–	–	–	–	–	–
Data packets analysis							0.5	1	1						
Communication channel load data				–			0.5	1	1			–			
DGU operation modes data							0.5	1	1						
Automation of testing							0.7	1	1						

The purposes of system testing also include the functions of checking the tools necessary for the Analyst to perform actions – analyzing the content of data exchange packets, traffic statistics of the information channel, operating schedules of power units and electrical loads. Their verification is possible only after a certain time of operation of the software in power system control mode and the suitability is the only functionality attribute that can be obtained as a result of this verification.

To interpret test results, all information messages and control actions are recorded in a protocol file. After testing is completed, the data from the protocol is analyzed, and on their basis, suitability (the proportion of correctly executed functions from all tested ones), interoperability (the proportion of correctly sent and processed requests) and accuracy for each component are calculated (the average value of the temporary inconsistency in the operation of protections for DGU protection). The obtained data is entered into the Table 1.

Weighting coefficients are determined for each component to calculate numerical values for suitability, interoperability, and accuracy. Next, using additive convolution and normalization, attribute values are calculated:

$$M = \frac{\sum_{i=1}^{15} k_{m.n.\mathrm{i}} \cdot k_{p.m.\mathrm{i}} \cdot \alpha_i}{\sum_{i=1}^{15} \alpha_i} \tag{1}$$

where, M is the value of the attribute; $k_{m.n.i}$ – the value of the test coverage; $k_{p.m.i}$ – test results; α_i is the weighting factor of the metric.

The next values of attributes were found for the software for AEPS monitoring and control based on the test results: suitability – 0.91; interoperability – 0.86; accuracy – 0.81. Based on the analysis of the results of all testing stages (Table 1), the conclusion can be made that it is possible to increase the value of the "suitability" and "interoperability" attributes by strengthening the testing of the system, and increasing the accuracy is associated with changing the software's work algorithms.

The value of the functionality is calculated according to a similar algorithm: attributes are assigned weighting factors and with the help of convolution and normalization, a numerical value is calculated. Setting the coefficients for suitability and interoperability as 1 and for accuracy as 0.8 gives the calculated value of functionality equal to 0.86.

2.3 Evaluation of practicality of the software

The assessment of practicality of the software consists in calculating of its three attributes: ergonomics, understandability, and efficiency of designing. For each of the attributes, sub-characteristics are also distinguished – indicators that are evaluated numerically. The methods for assessing practicality metrics were given in the study [23]. The most widely used among them testing, peer review and survey were identified. For the software of monitoring parameters and control AEPS, a survey was chosen as a basis for assessing practicality, an example for conducting which is given in Table 2. In this case, it is assumed that users (experts) will use the software by executing it to control the AEPS. This approach is dynamic testing, which is a process of testing software by running it to identify errors in production software and verify its functionality. Also, the important attribute indicators for this type of software are available in Table 2.

During software operation, errors can occur randomly, that is, initially their number, the moment and frequency of occurrence are uncertain. Error detections themselves collectively represent a flow of software failures, and failure information can in some sense be viewed as a flow of random numbers. The expert's scores who have different experience and expertise in a given subject area can also differ significantly, and it is not known in advance what assessment the expert will give to this or that attribute of practicality. Therefore, the methods of statistical analysis can be used to process the data obtained as a result of assessing practicality and testing.

In the survey, 10 users were involved to determine the practicality indicators. Based on the data obtained as a result of the survey, using the method of summarizing and grouping the results of statistical observation, the normalized average statistical values of practicality indicators were determined:

$$S_j = \frac{1}{k} \cdot \frac{\sum_{i=1}^{m} \alpha_{i\,j}}{m} \tag{2}$$

where, k – normalization coefficient (maximum value is 10); S_{ij} – evaluation of the j-th indicator by the i-th user; m_j – number of surveyed users.

Next, with the help of the introduction of weighting coefficients of importance and additive convolution, the calculation of attribute values is made:

$$A_j = \frac{1}{\sum_{i=1}^{m_j} p_{i\,j}} \cdot \sum_{i=1}^{m_j} \left(S_{i\,j} \cdot p_{i\,j}^{(n)} \right) \tag{3}$$

where, $p_{i\,j}^{(n)}$ – the weight coefficient of the i-th indicator of the j-th attribute; S_{ij} – normalized average statistical value of the i-th indicator of the j-th attribute; m_j – the number of indicators of the j-th attribute.

In a similar way (by entering importance coefficients and using additive convolution) one can get the numerical value of practicality attributes. The results of the calculations are summarized in Table 2.

To visualize the calculation results, it is suggested to use a linear bar chart [24], with which it is available to display the "coverage" of the requirements for indicators (Figure 2).

For this purpose, it is necessary to select the attributes with the most significant weights, and then among the selected ones – with the smallest calculated values:

$$\min_{A_j} \max_{p_j} \left\langle A_j,\, p_j^{(a)} \right\rangle \tag{4}$$

and then to choose the minimal in value of weight indicators of the attributes that were chosen at the previous stage:

$$\min_{S_{i\,j}} \max_{p_{i\,j}} \left\langle S_{i\,j},\, p_{i\,j}^{(n)} \right\rangle \tag{5}$$

The minimax criterion, which is used here, is one of the criteria for decision-making under conditions of uncertainty. As noted earlier, the process by which software errors occur is uncertain, and the moment of time of their occurrence can only be approximately estimated. Therefore, this criterion is used to make a decision, the aim of which is to find the best solution that maximizes the quality of the software and minimizes possible losses due to errors. This means that the decision maker cannot face a worse outcome than the one being oriented towards. Such approach weakens the influence of the subjective factor that inevitably appears when using the expert assessment method (i.e., a survey).

Table 2. The results of data processing to calculate the practicality of the software

Attributes and Indicators	Average Normalized Score	Importance/Significance Weighting		Attribute Value	Augmented Values	
		Indicator	Attribute		Attribute	Indicator
1. Ergonomics	–	–	1.00			–
1.1. Self-explanatory interface	0.82	0.90				0.90
1.2. Ease of performing frequent operations	0.84	1.00				0.84
1.3. Ease of complex operations	0.74	1.00		0.84	0.84	0.74
1.4. Acceptability of delays	0.92	0.90	0.84			1.01
1.5. Image of graphical components	0.84	0.80				1.01
1.6. The completeness of information on secondary dialog windows	0.88	0.90				0.97
2. Clarity	–	–	0.90			–
2.1. Intuitiveness	0.82	1.00				0.82
2.2. Compliance of software behaviour with expected	0.94	0.90	0.90	0.81	0.89	1.03
2.3. Ease of use of the component library	0.95	0.80				1.14
3. Learning Efficiency	–	–	0.90			–
3.1. Simplicity of mimic re-creation	0.88	1.00		0.76	0.83	0.88
3.2. Ease of recovery of software skills	0.84	0.90	0.84			0.92
3.3. Convenience of determining incorrect settings	0.80	1.00				0.80
Result				0.86		

The outcome of the described series of steps will be a set of indicators in the direction of which software improvement is the crucial. To ease the calculations of the minimax criterium, it is possible to calculate the values of attributes and indicators, considering the "inverse" weight, calculated as an algebraic complement of 2 "true" weights:

$$A_j^* = \left(2 - p_j\right) \cdot A_j \tag{6}$$

$$S_{i\,j}^* = \left(2 - p_{i\,j}\right) \cdot S_{i\,j} \tag{7}$$

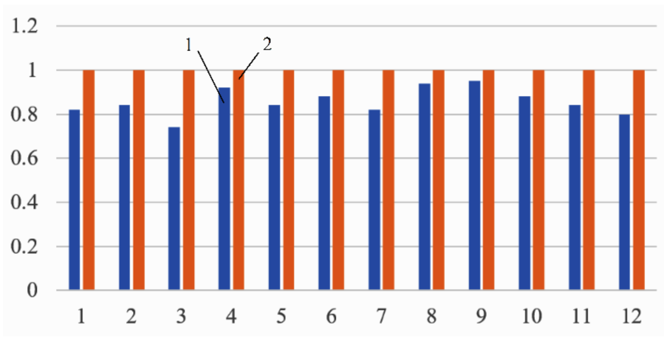

Figure 2. "Covering" requirements for practicality indicators (1 – indicators of the current version of the user interface; 2 – ideal indicators)

Then the determination of the most critical indicators for improving the practicality of the software will correspond to the procedure for determining the smallest values of the supplemented attributes, and then – among the smallest values of the supplemented indicators:

$$\min_{p_{i\,j}} \min_{A_j} \left\langle A_j^*, \left\{ S_{i\,j}^* \right\} \right\rangle \tag{8}$$

Next, the calculation results analysis of the practicality of the software for monitoring and managing the power system is considered. After calculations, a diagram of "coverage" of the requirements and a numerical value of practicality – 0.86 were obtained. For example, this value and coverage were found to be unsatisfactory, so it was decided to make changes in the software. The scope of changes (more specifically, practicability indicators) must be determined, considering the importance of individual directions. To do this, the largest attribute weight coefficient is selected – 1.00, which is decoded as "ergonomics is the most important for this software". Next, from the weigh coefficients of indicators that are elements of ergonomics, the largest one is chosen – 1.00. Two indicators match to the specified weight – "easiness of performing regular operations" and "easiness of performing complicated operations". From the latest, the one with the lowest average statistical value obtained as a result of the survey ("easiness of performing complicated operations") is chosen – the changes of the software are most important in this direction. However, to meet the requirements of practicality, it is necessary to change a few interdependent indicators, so it is possible, excluding the obtained indicator, to conduct the given approach again. The result for the situation under consideration will be the indicator "convenience of determining incorrect data exchange settings" – modifications in this regard will likewise have a substantial impact on enhancing practicality.

The indicated series of steps can be streamlined by performing calculations. the supplemented attribute and indicator values (values are "supplemented" to minimize the influence on the result of attributes and indicators having low weights). After their calculation, the smallest supplemented value of the attribute – 0.83 ("mastering efficiency" attribute) and the smallest supplemented indicator value is 0.8 ("ease of determining inaccurate data exchange settings" indicator) are selected for the corresponding attribute. A similar procedure is conducted to determine the second by the need for direction changes, but with the exclusion of the attribute "effectiveness of development". The outcome of the latter will yield the indicator "ease of performing complicated operations".

After selecting the indicators in the direction of which priority changes must be made, the interface of the software is adjusted. Then the survey is carried out one more time, and its results are processed in the given way. This process is repeated until the requirements and practicality values achieve a satisfactory level of "coverage". Thus, the calculated values of the metrics for various software quality attributes, obtained using the peer review technique, complement the results presented in the studies [25, 26] in terms of their use for software quality assessment for the ACS AEPS.

The attributes of mobility encompass adaptability, flexibility in deployment and replaceability.

Software reliability comprises three key attributes: fault tolerance, error tolerance, and ability to recover.

Indicators of the software failure tolerance are the mean square deviation and mathematical expectation of the uptime before the failure occurs, the conditional reliability function (the probability that the random uptime before the next failure will be greater than the specified one) [26]. The computation of these indicators relies on the time intervals between consecutive failures obtained as a result of functional testing (Table 3).

Table 3. Software failure statistics

Failure Number	1	2	3	4	5	6	7	8	9	10
Time Between Failures, Hours	0.08	1	4	7	9	10	12	12	14	20

Mathematical expectation, root mean square deviation and conditional reliability function are determined using statistical estimates of numerical properties related to the random time between failures:

$$m^*_{\Delta t} = \frac{1}{n_n}\sum_{i=1}^{n_n}\Delta t^{(i)} = \frac{1}{10}\cdot 89.08 = 8.908 \text{ (h)},$$
$$\left[\sigma^2_{\Delta t}\right]^* = \frac{1}{n_n - 1}\sum_{i=1}^{n_н}\left[\Delta t^{(i)} - m^*_{\Delta t}\right]^2 = \frac{1}{9}\cdot 33.75 = 3.75 \quad (9)$$

Mathematical expectation of the next (in this example, the eleventh) failure:

$$m_{t_n} = m_{\Delta t}\frac{n\cdot(n+1)}{2} = 8.908\cdot 66 \approx 588 \quad (10)$$

mean square deviation of the working time:

$$\sigma_{t_n} = \sigma_{\Delta t}\sqrt{\frac{1}{6}n\cdot(n+1)\cdot(n+2)} = \sqrt{3.75}\cdot\sqrt{7.7} = 5.37, \quad (11)$$

conditional reliability function:

$$p^{(n)}(\tau) = 0.5 - \Phi\left(\frac{\tau - n\cdot m_{\Delta t}}{\sigma_{\Delta t}\cdot\sqrt{n}}\right) = 0.5 - \Phi\left(\frac{\tau - 97.988}{6.423}\right), \quad (12)$$

where, τ is the specified earnings; $\Phi(u)$ is the probability integral.

As an example of the use of the reliability function, we consider a period of time, the probability of which is more than 85%.

$$0.85 = 0.5 - \Phi\left(\frac{\tau - 97.988}{6.423}\right);\ \tau \approx 91.3 \text{ (h)}. \quad (13)$$

Hence, with a probability of 85%, it can be said that the software will operate with no failures for over 91.3 hours. It is possible to increase this value due to the further collection of failures statistical data, which will lead to the reevaluation of statistical quality metrics.

The software error tolerance pertains to its capability to execute functions in abnormal conditions. In order to identify abnormal conditions, it is essential to highlight cases that require the use of third-party software and/or hardware. Table 4 presents the chosen instances, their requirements for the third-party tool availability and the reaction of the software for AEPS monitoring and control when errors occur.

Recoverability of the software for AEPS monitoring and control consists in restoring the mnemonic scheme and settings of its components. This requirement is a fundamental criterion for the software, and it is evaluated during functional testing.

Table 4. Software reaction to abnormal conditions

User Action	Software or/and Hardware	Software Reaction to Malfunctions
Precedents related to data exchange	Automation hardware	Indication of the the lack of communication and request of dequeuing
Predicting the result of connecting the load	MATLAB Engine	Informational message and automatic rejection of an action
Saving/restoring the mnemonic scheme and software environment; analysis of the communication channel congestion	Operating system tools for providing access to files	Information message
Accumulation and data analysis of the DGU modes and the loads state	DBMS PostgreSQL	Information message: the ability to change settings; table content atomicity

3. CONCLUSIONS

An analysis of methods for evaluating software quality characteristics was conducted, based on which it was proposed to use testing to evaluate functionality, an expert survey to evaluate practicality, and analytical methods to evaluate the remaining characteristics (supportability, portability, reliability, and resource efficiency). A formal approach to the determination of software quality indicators, particularly the usability, allowed to obtain quantitative characteristics of quality attributes that can be used throughout the entire software life cycle to ensure its support and development and serve as a kind of feedback during refactoring. Analytical expressions were obtained, and the values of individual characteristics and attributes of the software quality were calculated, which allowed to draw the following conclusions:

1. Increasing functionality of the software is connected with the enhancing system testing (increasing the suitability and

ability to interact) and modernization of the interaction algorithms of hardware and software automation tools (increasing accuracy).

2. Increasing reliability of the software requires further studies of the system for the occurrence of failures and faults. The further direction of research aimed at increasing the reliability of software, according to the author, lies in the presentation of algorithms of functioning of the software components in the form of digital finite state machines. The use of an automatic approach and the apparatus of pattern networks in the development and implementation of the software makes it quite easy altering the behavior of system components (by introducing new states, input and output actions into automaton) and incorporating new components (by constructing new constituents with the necessary connection to the pattern network). In this case, the algorithms can be described using state chart diagrams, in which all possible states of the component and the conditions for transitions between states are clearly identified. Such description of a component represents its discrete-deterministic model. This approach will allow developers to present the software architecture for controlling AEPS in the form of a network of digital finite state machines. Moreover, at each moment of time, the state of the system will be uniquely determined by the set of states of its elements. Using the theory of digital automata to solve this problem will allow developer to use the formal approach to software design, and at the testing stage, cover all possible states and transitions with tests, since their number will be finite. In addition, it becomes possible to implement a software “observer” that analyzes the current state of the system and compares it with the required one, and if a discrepancy between these states is detected, forcefully change the current state of the system (in this case, software) or some of its components to avoid its malfunctioning.

3. The practicality of the software can be increased by modernizing the means of determining improper data exchange settings and streamlining the sequence of user actions when doing complicated operations (managing load distribution during parallel operation of diesel generators, analyzing information flows, etc.). Presenting the results of the practicality assessment using a single number does not reveal possible problem areas, but it allows at various stages of development and implementation of the software to determine the necessity for further changes and the results of their implementation. It is possible to specify the direction of priority changes in the software by considering the results of calculations in the "reverse" direction. This will have a positive impact on the development process of autonomous power plant control systems and allows to improve the quality of the software for AEPS.

4. Time efficiency of the software depends on the topology and composition of the electric power system and the intensities of data flows from the software components. When the user interacts with the software, the components of the mnemonic scheme, which are used to control the elements of AEPS such as DGU and automation means, generate a data flow that is transmitted in real time over the network. Therefore, the congestion of the communication channel directly depends on the topology of the AEPS and the number of components used in the mnemonic scheme, and its characteristics should be calculated separately for each ACS AEPS control system.

REFERENCES

[1] Almogahed, A., Omar, M., Zakaria, N.H. (2019). Categorization refactoring techniques based on their effect on software quality attributes. International Journal of Innovative Technology and Exploring Engineering, 8(8S): 439-445.

[2] Vasilyevich, O.B. (2020). On the quality indicators of automated control systems software. Software Systems and Computational Methods, 2: 22-36. https://doi.org/10.7256/2454-0714.2020.2.28814

[3] Izzat, S., Saleem, N.N. (2023). Software testing techniques and tools: A review. Journal of Education and Science, 32(2): 30-44. https://doi.org/10.33899/edusj.2023.137480.1305

[4] Chevuturu, A., Mathur, D., Kumar, B., Charanya, R. (2022). A comparative survey on software testing tools. International Journal of Engineering and Advanced Technology, 11(6): 32-40. https://doi.org/10.35940/ijeat.F3664.0811622

[5] Kaur, A. (2020). A systematic literature review on empirical analysis of the relationship between code smells and software quality attributes. Archives of Computational Methods in Engineering, 27: 1267-1296. https://doi.org/10.1007/s11831-019-09348-6

[6] Ali, A.Q., Sultan, A.B.M., Ghani, A.A.A., Zulzalil, H. (2019). Empirical studies on the impact of software customization on quality attributes: A systematic review. Journal of Theoretical and Applied Information Technology, 97(6): 1747-1763.

[7] Lyaskovskiy, V.L., Bresler, I.B., Alasheev, M.A. (2021). Methodological and software tools for selecting solutions for the creation (development) of automated control systems. H&ES Research, 13: 48-59. https://doi.org/10.36724/2409-5419-2021-13-3-48-59

[8] Mahmoud, M.S., Ushkarenko, O., Dorogan, O. (2018). Research on information channel characteristics of a ship electric power system. Przeglad Elektrotechniczny, 94(6): 19-26. https://doi.org/10.15199/48.2018.06.04

[9] Al-Suod, M.M., Ushkarenko, A., Dorogan, O. (2018). Marine electric generating plants control systems software functional testing. International Journal of Computers, 3: 81-84.

[10] Hovorushchenko, T., Medzatyi, D., Voichur, Y., Lebiga, M. (2022). Method for forecasting the level of software quality based on quality attributes. Journal of Intelligent & Fuzzy Systems, 44(3): 3891-3905. https://doi.org/10.3233/JIFS-222394

[11] Shin, K., Thant, K.S., Tin, H.H.K. (2023). The impact of manual and automatic testing on software testing efficiency and effectiveness. Indian Journal of Science and Research, 3(3): 88-93.

[12] Tang, S., Zhang, Z., Zhang, Y., Zhou, J., Guo, Y., Liu, S., Guo, S., Li, Y., Ma, L., Xue, Y., Liu, Y. (2023). A survey on automated driving system testing: Landscapes and trends. ACM Transactions on Software Engineering and Methodology, 32(5): 1-62. https://doi.org/10.1145/3579642

[13] Soares, M.M., Rebelo, F., Ahram, T.Z. (Eds.). (2022). Handbook of Usability and User Experience: Methods and Techniques. CRC Press, Boca Raton.

[14] Eisty, N.U., Carver, J.C. (2022). Developers perception of peer code review in research software development. Empirical Software Engineering, 27: 1-26.

https://doi.org/10.1007/s10664-021-10053-x

[15] ISO/IEC 25010:2011. (2017). Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models, 34.

[16] ISO/IEC 25040:2011. (2019). Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Evaluation process. SC 7 System and Software Engineering, 45.

[17] Galin, D. (2018). Software Quality: Concepts and Practice. 1st ed., Wiley-IEEE Computer Society Press, Piscataway, NJ, 680. https://doi.org/10.1002/9781119134527

[18] Botchway, I.B., Emmanuel, A.A., Solomon, N., Kayode, A.B. (2021). Evaluating software quality attributes using Analytic Hierarchy Process (AHP). International Journal of Advanced Computer Science and Applications, 12(3): 165-173. https://doi.org/10.14569/IJACSA.2021.0120321

[19] Munoz, D.J., Pinto, M., Fuentes, L. (2019). HADAS: Analysing quality attributes of software configurations. In Proceedings 23rd International Systems and Software Product Line Conference, Paris, France, pp. 13-16. https://doi.org/10.1145/3307630.3342385

[20] Dougherty, R.E., Kleine, K., Wagner, M., Colbourn, C.J., Simos, D.E. (2022). Algorithmic methods for covering arrays of higher index. Journal of Combinatorial Optimization, 45(1): 1-21. https://doi.org/10.1007/s10878-022-00947-x

[21] Dhivya, D., Nirmala, K. (2018). Study on integration testing and system testing. International Journal of Creative Research Thoughts, 6(2): 794-798.

[22] Lotfi, Z., Khalifi, H., Ouardi, F. (2023). Efficient algebraic method for testing the invertibility of finite state machines. Computation, 11(125): 1-17. https://doi.org/10.3390/computation11070125

[23] Bayomi, H., Sayed, N.A., Hassan, H., Wassif, K. (2022). Application-based usability evaluation metrics. International Journal of Advanced Computer Science and Applications, 13(7): 84-91. https://doi.org/10.14569/IJACSA.2022.0130712

[24] Waldner, M., Diehl, A., Gračanin, D., Splechtna, R., Delrieux, C., Matković, K. (2020). A comparison of radial and linear charts for visualizing daily patterns. IEEE Transactions on Visualization and Computer Graphics, 26(1): 1033-1042. https://doi.org/10.1109/TVCG.2019.2934784

[25] Almogahed, A., Omar, M., Zakaria, N.H., Muhammad, G., AlQahtani, S.A. (2023). Revisiting scenarios of using refactoring techniques to improve software systems quality. IEEE Access, 11: 28800-28819. https://doi.org/10.1109/ACCESS.2022.3218007

[26] Rawat, S., Rawat, R.S., Ram, M. (2015). A review on software reliability: Metrics, models and tools. Mathematics in Engineering. Science and Aerospace, 6(2): 135-156.